Replication of Decentralized Learning over Wireless Networks with Broadcast-Based Subgraph Sampling

Jedidi Imene
EURECOM
jedidi@eurecom.fr

Khachmadi Arsalene
EURECOM
khachmadi@eurecom.fr

Rosato Sara
EURECOM
rosato@eurecom.fr

***Abstract*—This report presents a duplication of the BASS (Broadcast-based Subgraph Sampling) [1] framework to improve decentralized learning over wireless networks using consensus-based decentralized stochastic gradient descent (D-SGD). Exploring the broadcast nature of wireless networks and probabilistic subgraph sampling, BASS optimizes model update transmissions. We implemented the framework using the original parameters specified in the paper. By activating non-interfering node subsets with probabilities based on network connectivity importance, BASS achieves faster convergence under communication cost constraints. Our results confirm that BASS significantly outperforms traditional methods, enhancing convergence efficiency in decentralized learning scenarios.**

—

Keywords: Decentralized learning, Wirless networks, Topologies, Communication

I. Introduction & state of art

This report presents the implementation, validation, and replication of the results obtained in the paper "Decentralized Learning over Wireless Networks with Broadcast-Based Subgraph Sampling" by Daniel Pérez Herrera, Zheng Chen, and Erik G. Larsson [1]. The project's objective is to replicate the proposed BASS (BroAdcast-based Subgraph Sampling) framework, which aims to enhance the efficiency of decentralized stochastic gradient descent (D-SGD) in wireless networks.

Decentralized learning involves multiple agents collaboratively training a machine learning model using their local data without sharing the data itself. A commonly used optimization method in this context is Consensus-Based D-SGD, where agents update their models by integrating information from their neighbors. However, current methods often overlook the significant communication costs and delays inherent in wireless networks, caused by issues such as packet collision and access control.

The BASS framework addresses these challenges by utilizing the broadcasting capabilities of wireless channels to improve the efficiency of information dissemination. Key innovations of BASS include broadcast transmission, which leverages the broadcast nature of wireless communication to disseminate information more efficiently; subgraph sampling, which activates random subsets of non-interfering nodes to broadcast model updates to their neighbors, enhancing network connectivity; and symmetric communication, which maintains bi-directional links during the consensus update to ensure symmetric communication.

Further innovations include subgraph partitioning, where the network topology is divided into collision-free subsets using a vertex-coloring algorithm, allowing simultaneous transmissions without interference within each subset. Probabilistic scheduling assigns higher activation probabilities to subsets containing critical nodes, optimizing communication costs and ensuring efficient information exchange. Additionally, the weight matrix is optimized to accelerate convergence by solving an optimization problem that minimizes communication costs.

BASS offers several advantages, including improved convergence, achieving faster results compared to existing link-based scheduling methods by activating more links within the same number of transmission slots. The method is scalable and resilient to single-node failures, making it suitable for large and dynamic network topologies.

The performance of BASS was evaluated through simulations involving the training of a multi-layer perceptron on the MNIST dataset across various network topologies. The results demonstrated significant improvements in test accuracy per transmission slot compared to both a modified version of MATCHA and the full communication case.

In conclusion, BASS effectively leverages broadcast-based communication to enhance the efficiency of decentralized learning over wireless networks. The framework shows great potential for applications with constrained communication budgets, offering a robust and scalable solution for decentralized training.

II. Related work

Full communication in the context of decentralized stochastic gradient descent (D-SGD) refers to a scenario where every node (or agent) in the network communicates with all other nodes directly and at every iteration. This means that during each update step, each node sends its model parameters to every other node and receives model parameters from every other node, ensuring complete information sharing across the network. While this approach can lead to faster convergence due to the rich exchange of information, it also incurs high communication overhead, especially in large or dense networks. This overhead includes increased communication delays and higher energy consumption, which are critical considerations in wireless networks.

Another technique is MATCHA (Matching Decomposition Sampling) [2], a decentralized SGD algorithm designed to address the error-runtime trade-off typically encountered in decentralized training. This trade-off arises because denser network topologies while facilitating faster error convergence, incur higher communication delays per iteration.
MATCHA aims to balance this by decomposing the communication topology into matchings, thereby allowing parallel inter-node communication and balancing communication time and error convergence by adjusting the communication budget. It focuses on reducing runtime while maintaining convergence speed through flexible communication budgets and matching decomposition.

This strategy is different from BASS, which leverages broadcast-based communication, using subgraph sampling to activate non-interfering nodes that broadcast model updates, improving efficiency and reducing delays.

MATCHA demonstrates significant reductions in wall-clock time for training tasks, showing up to a 5.2× improvement in runtime. BASS shows performance gains in test accuracy per transmission slot and is resilient to single-node failures. MATCHA is ideal for environments where communication costs are critical but not the only concern, while BASS is optimized for scenarios with constrained communication budgets, addressing issues like packet collision and access control in wireless networks.

III. Implementation Details

A. Libraries and Tools Used

1) Python: The primary programming language used for the implementation and experimentation.

2) NetworkX: A comprehensive library for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.
Usage:

- **Graph Creation**: Used to create and manage the network topology, adding nodes and edges to represent the communication network.
- **Centrality Calculation**: Utilized for calculating betweenness centrality to determine the importance of nodes in the network.
- **Graph Modifications**: Helps in modifying the network by adding auxiliary links to improve connectivity and ensure collision-free subsets.

3) Matplotlib: A plotting library for creating static, animated, and interactive visualizations in Python.
Usage:

- **Visualization**: Used for visualizing the network topology and the modifications made to it.
- **Graph Drawing**: Helps in drawing and displaying the network graphs with labels, node sizes, colors, and edge styles.

4) TensorFlow/Keras: An open-source platform for machine learning, and Keras is its high-level API for building and training deep learning models.
Usage:

- **Data Loading**: Used to load and preprocess the MNIST dataset for training and testing the neural network model.
- **Model Definition**: Helps in defining and compiling a neural network model for the image classification task.
- **Training and Evaluation**: Used to train the model on the training data and evaluate its performance on the test data.

B. Network Topology

In this section, we describe how the network topology was created, visualized, and utilized for implementing the BASS framework.

C. Network Topology

Graph Creation: An empty graph `G` was created using NetworkX.

Adding Nodes: Nodes were added to the graph using the `add_nodes_from` method.

Adding Edges: Edges were added to connect the nodes, forming the network topology.

Visualization: The network topology was visualized using Matplotlib. The nodes are labeled and colored in sky blue, while the edges are colored in black.

Visualization of the Initial Network Topology: The initial network topology was visualized as shown below in fig. 1. This visualization helps in understanding the structure of the network and the connections between the nodes.

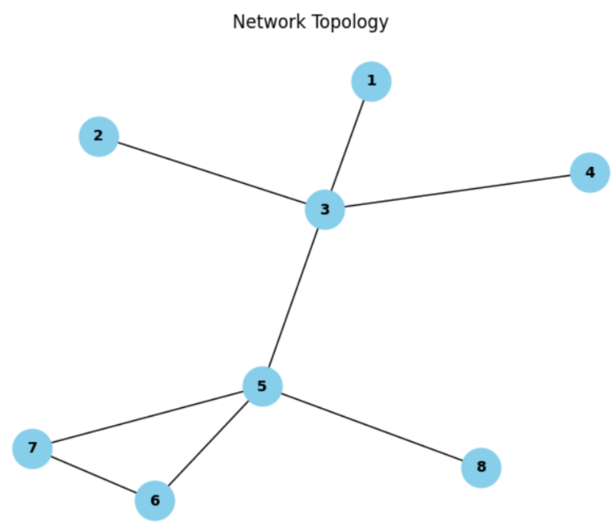

Fig. 1. Network topology

The network topology shows 8 nodes with the following connections:

- Node 1 is connected to Node 3.
- Node 2 is connected to Node 3.

- Node 3 is connected to Nodes 1, 2, 4, and 5.
- Node 4 is connected to Node 3.
- Node 5 is connected to Nodes 3, 6, 7, and 8.
- Node 6 is connected to Nodes 5 and 7.
- Node 7 is connected to Nodes 5 and 6.
- Node 8 is connected to Node 5.

By constructing and visualizing this network topology, we established the basis for further calculations and modifications necessary for implementing the BASS framework. This topology will be used for calculating betweenness centrality, determining sampling probabilities, and making modifications to ensure collision-free subsets.

D. *Calculation of node importance*

Betweeness centrality: To calculate the importance of each node, we use betweenness centrality, which measures the number of shortest paths passing through each node. This centrality metric is crucial for determining the sampling probabilities for each node.

Explanation

- **Betweenness Centrality Calculation**: The betweenness centrality for each node in the graph was calculated using NetworkX's `betweenness_centrality` function. This function computes the shortest paths for all node pairs and determines how frequently each node lies on these paths.
- **Importance of Nodes**: Nodes with higher betweenness centrality values are considered more important because they act as bridges in the network, facilitating communication between different parts of the network.

Normalization and Sampling Probabilities: The betweenness centrality values were then normalized to obtain sampling probabilities for each node. This ensures that nodes with higher centrality have a higher chance of being selected for communication.

Explanation

- **Epsilon Adjustment**: To avoid zero sampling probabilities, we add a small epsilon value to each betweenness centrality score. This ensures that every node has a non-zero probability of being sampled.
- **Normalization**: The betweenness centrality values were summed up, and each node's centrality was divided by this sum to obtain the sampling probability.
- **Sampling Probabilities**: The resulting probabilities were printed for each node, showing the likelihood of each node being selected for communication based on its importance.
  By calculating and normalizing the betweenness centrality values, we were able to assign appropriate sampling probabilities to each node, reflecting their importance in the network. This step is crucial for the BASS framework, as it ensures that more critical nodes are more frequently involved in the communication process, thereby improving the efficiency and convergence of the decentralized learning algorithm.

E. *Network Modifications*

In this part, we construct an auxiliary graph by adding links between nodes that share common neighbors, enhancing the original topology to reflect potential communication scenarios.

Auxiliary Links Addition: This auxiliary graph helps in ensuring collision-free broadcasts during the transmission slots.

Explanation

- **Graph Duplication**: A new graph `H` was created by duplicating the original graph `G` to allow modifications.
- **Neighbor and Common Neighbor Identification**: For each node, the neighbors and common neighbors were identified to form subgraphs.
- **Auxiliary Links**: Edges were added between nodes and their common neighbors to create auxiliary links, ensuring improved connectivity and collision-free communication.

Visualization of Modified Network Topology: The modified network topology was visualized as shown below. This visualization helps in understanding the new structure of the network and how auxiliary links were added to enhance connectivity.

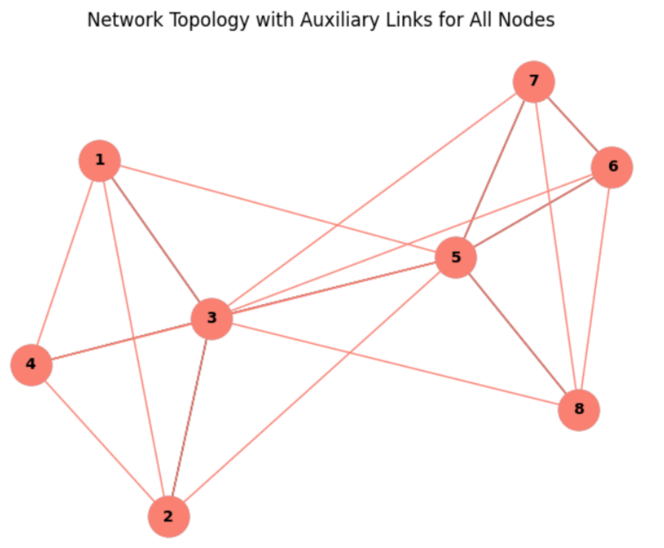

Fig. 2. Network topology with Auxiliary

F. *Apply Greedy Coloring Algorithm to Find Subsets*

Greedy Coloring Algorithm: The greedy coloring algorithm is used to assign colors to the nodes of the Auxiliary Graph such that no two adjacent nodes have the same color. This algorithm helps in creating collision-free subsets of nodes that can transmit simultaneously without interfering with each other.

Explanation

- **Greedy Coloring Algorithm**: The `greedy_color` function from NetworkX was used to assign colors to the nodes of the auxiliary graph `Ga`. The strategy `largest_first` was chosen to color the nodes in order of decreasing degree.

- **Subset Creation**: Nodes were grouped into subsets based on their assigned colors. Each subset represents a group of nodes that can transmit simultaneously without collision.
- **Subset Printing**: The resulting subsets were printed to verify the partitioning.

Visualization of Colored Graph: The graph was visualized to show the coloring and the resulting collision-free subsets. This visualization helps in understanding how the nodes were partitioned.

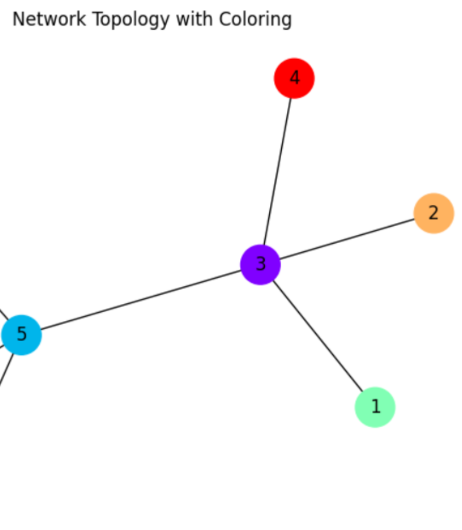

Fig. 3. Coloring algorithm applied to the Network Topology

top

Subsets of Nodes: The resulting subsets of nodes, based on their colors, are as follows:

- Subset 0: [3]
- Subset 1: [5]
- Subset 2: [1, 6]
- Subset 3: [2, 7]
- Subset 4: [4, 8]

By applying the greedy coloring algorithm, we effectively partitioned the network into collision-free subsets. This step ensures that nodes within each subset can broadcast their model updates simultaneously without causing interference, thus improving the communication efficiency of the BASS framework.

G. Partial Communication with Probabilistic Scheduling

Betweenness Centrality and Sampling Probabilities of Subsets: After forming the subsets, we calculate the betweenness centrality for each subset to determine its overall importance. We define the betweenness centrality of a subset as:

$$b_{S_j} = \sum_{i=1}^{N} b_i \mathbb{1}_{\{i \in V_j\}}, \tag{1}$$

where b_i is the betweenness centrality of node i, V_j is the set of nodes in subset S_j, and $\mathbb{1}_{\{i \in V_j\}}$ is the indicator function that is 1 if $i \in V_j$ and 0 otherwise. The sampling probabilities for each subset are then derived based on these centrality scores, ensuring that more critical subsets are activated more frequently. We choose the probability of each sample and it is giving by:

$$p_{S_j} = \min\{1, \gamma b_{S_j}\}, \tag{2}$$

where γ is a scaling factor chosen such that the communication budget is equal to the average number of transmission slots per iteration:

$$\beta = \sum_{i=1}^{q} p_{S_i} \tag{3}$$

Designing Scheduling Probabilities: The final step involves designing the scheduling probabilities for each subset based on their betweenness centrality. By assigning higher probabilities to subsets with greater importance, we optimize the communication schedule to accelerate the convergence of the D-SGD algorithm. We achieve this by first computing the total probability as the sum of the betweenness centrality values for all subsets. Next, we calculate γ by dividing this total probability by the communication budget, which represents the average number of transmission slots available per iteration. Using this scaling factor, we assign scheduling probabilities to each subset by multiplying γ with the betweenness centrality of the subset and capping the value at 1 to ensure valid probability values. This method respects the communication budget while prioritizing subsets with higher betweenness centrality, thereby enhancing the overall efficiency of the communication process in the decentralized learning network. By completing this part, we trained a Multilayer Perceptron (MLP).

H. MLP implementation

This code presents an implementation of a Multilayer Perceptron (MLP) using TensorFlow's Keras API. It's designed to be a flexible and configurable framework for training and evaluating neural networks. The key components include:

- **Model Architecture:** The model is built using the Sequential API with the specified input shape. Two dense layers with 256 and 128 neurons respectively are added, both utilizing ReLU activation functions. Dropout layers with a dropout rate of 0.5 are introduced after each dense layer to prevent overfitting. Regularization with L2 regularizer is applied to the kernel weights of the dense layers using the specified `weight_decay` parameter.
- **Learning Rate Scheduler:** defines a function to adjust the learning rate based on the current epoch number. It decays the learning rate by a factor of `decay_rate` every `decay_steps` epochs.
- **Compilation**: The `compile` method compiles the model. It uses the Adam optimizer with a specified initial learning rate and weight decay. Sparse categorical cross-entropy is used as the loss function, suitable for integer target labels. Accuracy is tracked as a metric during training.

- **Training**: The `fit` method trains the model on input data and target labels for a specified number of epochs and with a specified batch size. It utilizes a learning rate scheduler callback to adjust the learning rate during training.
- **Evaluation:** evaluates the trained model on input data and target labels, returning the loss value and accuracy metric.

This implementation enhances the original MLP with dropout regularization, potentially improving its generalization capabilities.

IV. Experiments

In this section, we assess the effectiveness of our proposed broadcast-based subgraph sampling method for D-SGD over wireless networks. We conduct simulations using a multi-layer perceptron (MLP), whose implementation is explained in the section above, trained on the MNIST dataset, consisting of 60,000 images for training and 10,000 for testing.
For our training setup, we divide the training dataset into 2N shards, assigning each user 2 randomly selected shards without repetition. We employ the SparseCategoricalCrossentropy loss function and the Adam optimizer with a decaying learning rate, and we set the number of epochs to 40. During experimentation, we compare different weight decays, namely weight_decays = [0.0001, 0.0005, 0.001, 0.005], and find that the best-performing weight decay value is 0.0001. Each experimental iteration consists of 50 communication rounds. Throughout our experiments, we analyzed the performance improvement in test accuracy, and the results obtained in the base topology are

Test Loss: 0.0741
Test Accuracy: 99.19%

we implemented and tested our code on three distinct topologies to observe how each configuration affects the results. The three topologies considered are:

- Star-like Topology with 25 nodes (fig. 4)
- Linear Topology with 20 nodes (fig. 5)
- Tree-like Topology with 15 nodes (fig. 6)

TABLE I
Comparison of Test Loss and Accuracy for Different Graphs

Graph	Test Loss	Accuracy
1	0.1826	96.89%
2	0.1661	97.26%
3	0.1578	97.24%

V. Interpretation

From the results, we can conclude that:

- The **tree-like topology (Graph 3)** is the most effective for the given task, as it achieves the lowest test loss, indicating better generalization and model performance.

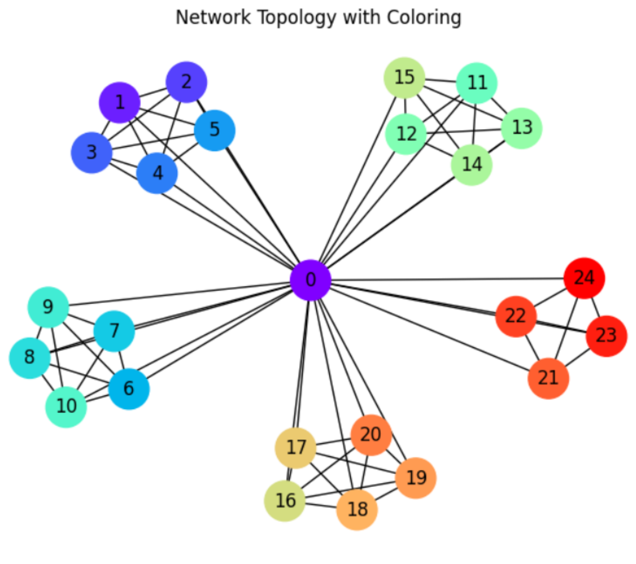

Fig. 4. Network topology with 25 nodes

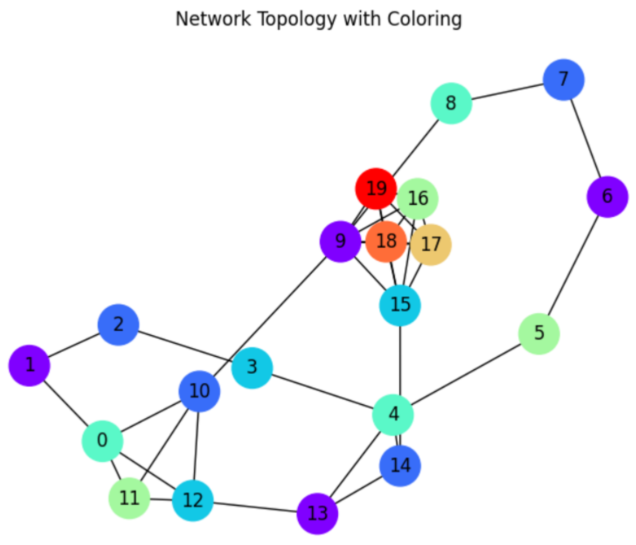

Fig. 5. Network topology with 20 nodes

- The **linear topology (Graph 2)** also performs very well, with the highest accuracy, suggesting that sequential data flow can be beneficial.
- The **star-like topology (Graph 1)**, while still performing well, has relatively higher test loss and lower accuracy, which could be due to the central bottleneck or over-clustering.

These insights suggest that for the given task, more distributed or hierarchical structures (like in Graph 2 and Graph 3) might provide better performance compared to centralized structures (like in Graph 1). This can inform decisions on network design and data distribution for similar machine learning tasks.

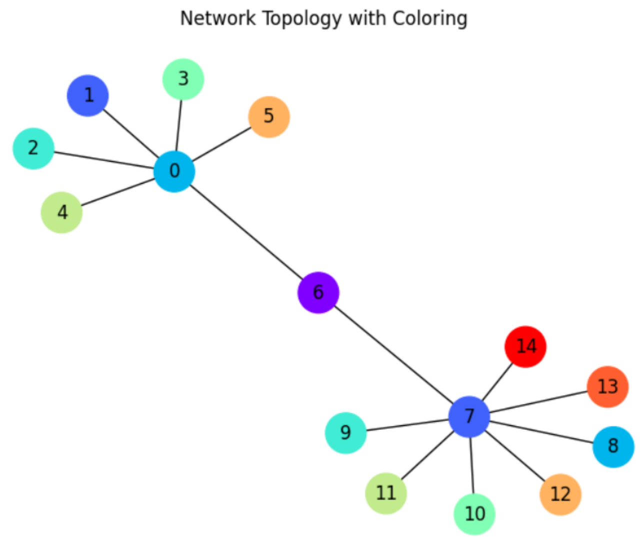

Fig. 6. Network topology with 15 nodes

VI. Alternative approach

In this experiment, we relied on betweenness centrality to assess node importance within the network. However, it's important to note that other methods, such as eigenvalue analysis or entropy measurement, offer alternative perspectives. Eigenvalue analysis explores the structural properties and connectivity patterns of the network, while entropy-based approaches uncover diversity and uncertainty present within the network's attributes or structures. By measuring the entropy of node attributes or network structures, one can gain insights into the distribution and variability of information or characteristics across the network. Higher entropy values indicate greater diversity or randomness, while lower entropy values suggest more uniformity or predictability. Integrating these methods alongside betweenness centrality can provide a more comprehensive understanding of node importance, enhancing the robustness and convenience of our analysis.

An alternative approach to using a coloring algorithm for partitioning the network into collision-free subsets is to employ a clustering algorithm like K-means or spectral clustering, which are widely used in network analysis for grouping nodes based on their connectivity patterns.

By leveraging the structure and connectivity of the network, these algorithms can provide robust solutions that enhance the efficiency of information transmission in wireless networks. Integrating clustering methods alongside traditional approaches can further enrich the analysis and optimization of network communication topologies.

VII. Conclusion

In this replication study, we use and reproduce BASS a broadcast-based subgraph sampling technique designed to accelerate decentralized learning in wireless networks with limited communication capabilities. Following the BASS technique, we divide the network structure into collision-free subsets, allowing nodes within each subset to simultaneously broadcast model updates to their neighbors. Using a probabilistic activation approach, these subsets are activated based on the contribution of nodes to overall graph connectivity. Our simulated findings validated the usefulness of BASS.

References

[1] D. P. Herrera, Z. Chen, and E. G. Larsson, "Decentralized learning over wireless networks with broadcast-based subgraph sampling," 2023.

[2] J. Wang, A. K. Sahu, Z. Yang, G. Joshi, and S. Kar, "Matcha: Speeding up decentralized sgd via matching decomposition sampling," 2019.